

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

**Methods And Arrangements For Protecting
Information In Forwarded Authentication Messages**

Inventors:

John E. Brezak
Richard B. Ward

ATTORNEY'S DOCKET NO. MS1-648US

Methods And Arrangements For

Protecting Information In Forwarded Authentication Messages

TECHNICAL FIELD

This invention relates generally to computer access control, and more particularly to methods and arrangements for selectively protecting information in forwarded authentication messages.

BACKGROUND

Access control is paramount to computer security. To protect the integrity of computer systems and the confidentiality of important data, various access control schemes have been implemented to prevent unauthorized users and malicious attackers from gaining access to computer resources.

To ensure the comprehensiveness of computer security, access control is often implemented on various levels. For instance, on the level of one computer, a user is typically required to go through a logon procedure in which the computer determines whether the user is authorized to use the computer. In addition, on the level of a computer network, a user is commonly required to go through a user-authentication process for purposes of controlling the user's access to various network services. Even after a network access control server has authenticated the user, the user may still have to request a permit for a specific server in order to access that service. Various schemes based on different protocols, such as the Kerberos 5 protocol, have been proposed and implemented for controlling network access control by means of user authentication.

1 Generally, the user logon for a computer and the user authentication for
2 network access control are two separate procedures. Nevertheless, to minimize
3 the burden on a user in dealing with the different access control schemes, the user
4 logon and the user authentication for network access are sometimes performed
5 together. For example, in the case where the user authentication is implemented
6 under the Kerberos protocol, when the user logs on the computer, the computer
7 may also initiate a Kerberos authentication process. In the authentication process,
8 the computer contacts a Kerberos Key Distribution Center (KDC) to first obtain a
9 ticket-granting ticket (TGT) for the user. The computer can then use the TGT to
10 obtain from the KDC, a session ticket for itself.

11 As networks have evolved, there has been a trend to have multiple tiers of
12 server/service computers arranged to handle client computer requests. A simple
13 example is a client computer making a request to a World Wide Web website via
14 the Internet. Here, there may be a front-end web server that handles the formatting
15 and associated business rules of the request, and a back-end server that manages a
16 database for the website. For additional security, the web site may be configured
17 such that an authentication protocol forwards (or delegates) credentials and/or
18 possibly other information from the front-end server to the back-end server. This
19 practice is becoming increasingly common in many websites.

20 It appears that this forwarding/delegating practice will expand in the near
21 future to further include not only front-end and back-end websites, but also
22 websites that provide an aggregated view of other websites. One example is a
23 travel service site. Here, a client may be willing to allow the travel service site to
24 forward a personal travel profile to air carriers, car rental companies, hotel chains,
25 etc., but not to Bob's Pickpocket Service. Unfortunately, conventional

1 authentication schemes do not allow for selective forwarding on the part of the
2 client.

3 Consequently, there is a need for methods and arrangements that allow for
4 selective forwarding on the part of the client, of information associated with client.

5 **SUMMARY**

6 In accordance with certain aspects of the present invention, methods and
7 arrangements are provided to selectively control access to the authentication
8 information or portions thereof. The methods and arrangements are based on a
9 scheme wherein the authentication information further includes specially encoded
10 portions that can only be decoded by selected server-based services/processes.
11 One method for use in protecting information in forwarded authentication
12 messages includes encoding the selected data using an encryption key, then
13 encoding the encryption key itself, using at least one other encryption key that
14 only certain selected servers/services have access to, and then encapsulating the
15 resulting encoded data and the encoded encryption key in an authentication
16 message. This and other methods are particularly applicable to Kerberos and other
17 like authentication arrangements.

18 Additional features and advantages of the invention will be made apparent
19 from the following detailed description of illustrative embodiments, which
20 proceeds with reference to the accompanying figures.

21 **BRIEF DESCRIPTION OF THE DRAWINGS**

22 A more complete understanding of the various methods and arrangements
23 of the present invention may be had by reference to the following detailed
24 description when taken in conjunction with the accompanying drawings wherein:
25

Fig. 1 is a block diagram generally illustrating an exemplary computer system on which the present invention may be implemented.

Fig. 2 is a block diagram depicting an exemplary client-server environment.

Fig. 3 is an illustrative block diagram depicting an authentication message.

Figs 4-8 are block diagrams depicting exemplary Kerberos message exchanges.

Fig. 9 is an illustrative diagram depicting an exemplary improved authentication message in accordance with certain aspects of the present invention, and suitable for use in any of the configurations in Figs. 1-8.

DETAILED DESCRIPTION

Turning to the drawings, wherein like reference numerals refer to like elements, the invention is illustrated as being implemented in a suitable computing environment. Although not required, the invention will be described in the general context of computer-executable instructions, such as program modules, being executed by a personal computer. Generally, program modules include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Moreover, those skilled in the art will appreciate that the invention may be practiced with other computer system configurations, including hand-held devices, multi-processor systems, microprocessor based or programmable consumer electronics, network PCs, minicomputers, mainframe computers, and the like. The invention may also be practiced in distributed computing environments where tasks are performed by remote processing devices that are linked through a communications network. In

1 a distributed computing environment, program modules may be located in both
2 local and remote memory storage devices.

3 Fig.1 illustrates an example of a suitable computing environment 120 on
4 which the subsequently described methods and arrangements may be
5 implemented.

6 Exemplary computing environment 120 is only one example of a suitable
7 computing environment and is not intended to suggest any limitation as to the
8 scope of use or functionality of the improved methods and arrangements described
9 herein. Neither should computing environment 120 be interpreted as having any
10 dependency or requirement relating to any one or combination of components
11 illustrated in computing environment 120.

12 The improved methods and arrangements herein are operational with
13 numerous other general purpose or special purpose computing system
14 environments or configurations. Examples of well known computing systems,
15 environments, and/or configurations that may be suitable include, but are not
16 limited to, personal computers, server computers, thin clients, thick clients, hand-
17 held or laptop devices, multiprocessor systems, microprocessor-based systems, set
18 top boxes, programmable consumer electronics, network PCs, minicomputers,
19 mainframe computers, distributed computing environments that include any of the
20 above systems or devices, and the like.

21 As shown in Fig. 1, computing environment 120 includes a general-purpose
22 computing device in the form of a computer 130. The components of computer
23 130 may include one or more processors or processing units 132, a system
24 memory 134, and a bus 136 that couples various system components including
25 system memory 134 to processor 132.

1 Bus 136 represents one or more of any of several types of bus structures,
2 including a memory bus or memory controller, a peripheral bus, an accelerated
3 graphics port, and a processor or local bus using any of a variety of bus
4 architectures. By way of example, and not limitation, such architectures include
5 Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA)
6 bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA)
7 local bus, and Peripheral Component Interconnects (PCI) bus also known as
8 Mezzanine bus.

9 Computer 130 typically includes a variety of computer readable media.
10 Such media may be any available media that is accessible by computer 130, and it
11 includes both volatile and non-volatile media, removable and non-removable
12 media.

13 In Fig. 1, system memory 134 includes computer readable media in the
14 form of volatile memory, such as random access memory (RAM) 140, and/or non-
15 volatile memory, such as read only memory (ROM) 138. A basic input/output
16 system (BIOS) 142, containing the basic routines that help to transfer information
17 between elements within computer 130, such as during start-up, is stored in ROM
18 138. RAM 140 typically contains data and/or program modules that are
19 immediately accessible to and/or presently being operated on by processor 132.

20 Computer 130 may further include other removable/non-removable,
21 volatile/non-volatile computer storage media. For example, Fig. 1 illustrates a
22 hard disk drive 144 for reading from and writing to a non-removable, non-volatile
23 magnetic media (not shown and typically called a "hard drive"), a magnetic disk
24 drive 146 for reading from and writing to a removable, non-volatile magnetic disk
25 148 (e.g., a "floppy disk"), and an optical disk drive 150 for reading from or

1 writing to a removable, non-volatile optical disk 152 such as a CD-ROM, CD-R,
2 CD-RW, DVD-ROM, DVD-RAM or other optical media. Hard disk drive 144,
3 magnetic disk drive 146 and optical disk drive 150 are each connected to bus 136
4 by one or more interfaces 154.

5 The drives and associated computer-readable media provide nonvolatile
6 storage of computer readable instructions, data structures, program modules, and
7 other data for computer 130. Although the exemplary environment described
8 herein employs a hard disk, a removable magnetic disk 148 and a removable
9 optical disk 152, it should be appreciated by those skilled in the art that other types
10 of computer readable media which can store data that is accessible by a computer,
11 such as magnetic cassettes, flash memory cards, digital video disks, random access
12 memories (RAMs), read only memories (ROM), and the like, may also be used in
13 the exemplary operating environment.

14 A number of program modules may be stored on the hard disk, magnetic
15 disk 148, optical disk 152, ROM 138, or RAM 140, including, e.g., an operating
16 system 158, one or more application programs 160, other program modules 162,
17 and program data 164.

18 The improved methods and arrangements described herein may be
19 implemented within operating system 158, one or more application programs 160,
20 other program modules 162, and/or program data 164.

21 A user may provide commands and information into computer 130 through
22 input devices such as keyboard 166 and pointing device 168 (such as a "mouse").
23 Other input devices (not shown) may include a microphone, joystick, game pad,
24 satellite dish, serial port, scanner, camera, etc. These and other input devices are
25 connected to the processing unit 132 through a user input interface 170 that is

coupled to bus 136, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB).

A monitor 172 or other type of display device is also connected to bus 136 via an interface, such as a video adapter 174. In addition to monitor 172, personal computers typically include other peripheral output devices (not shown), such as speakers and printers, which may be connected through output peripheral interface 175.

Computer 130 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 182. Remote computer 182 may include many or all of the elements and features described herein relative to computer 130.

Logical connections shown in Fig. 1 are a local area network (LAN) 177 and a general wide area network (WAN) 179. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets, and the Internet.

When used in a LAN networking environment, computer 130 is connected to LAN 177 via network interface or adapter 186. When used in a WAN networking environment, the computer typically includes a modem 178 or other means for establishing communications over WAN 179. Modem 178, which may be internal or external, may be connected to system bus 136 via the user input interface 170 or other appropriate mechanism.

Depicted in Fig. 1, is a specific implementation of a WAN via the Internet. Here, computer 130 employs modem 178 to establish communications with at least one remote computer 182 via the Internet 180.

1 In a networked environment, program modules depicted relative to
2 computer 130, or portions thereof, may be stored in a remote memory storage
3 device. Thus, e.g., as depicted in Fig. 1, remote application programs 189 may
4 reside on a memory device of remote computer 182. It will be appreciated that the
5 network connections shown and described are exemplary and other means of
6 establishing a communications link between the computers may be used.

7 This description will now focus on certain aspects of the present invention
8 associated with protecting information in forwarded authentication messages.
9 While the following description focuses on exemplary Kerberos-based systems
10 and improvements there to, the various methods and arrangements of the present
11 invention are also clearly applicable to other authentication systems and
12 techniques.

13 In certain exemplary systems, the Kerberos protocol is implemented as a
14 Security Service Provider (SSP) that is accessible via a Security Support Provider
15 Interface (SSPI). In this manner, applications can directly access authentication
16 protocol services through SSPI. Those skilled in the art will recognize that other
17 services/systems may be used, such as, for example, an attribute-based system,
18 SSL, etc.

19 For example, Fig. 2 shows how applications or other programs may use
20 selected security interfaces, for example, SSPIs. Here, an exemplary system 200
21 is depicted as having a client machine 202 and server machine 204. Client
22 machine 202 is operatively coupled to SSP 208(a). Server machine 204 is
23 operatively coupled to SSP 208(b). In certain exemplary configurations, SSP
24 208a-b may include Kerberos related SSPI services, or the like.

1 In this example, the SSP 208(a-b) provides three security services, namely,
2 authentication, data integrity and data privacy. The authentication protocol uses
3 encryption to provide each of these services. Here, the operating system can
4 provide public key and/or secret key encryption. Preferably, the authentication
5 protocol will rely mostly on secret key encryption (also called private key,
6 symmetric, single key, or shared secret encryption).

7 Rather than require its users (known as principals) to invent special
8 encryption keys, the authentication protocol may, for example, just use ordinary
9 passwords. But those passwords aren't used to encrypt the actual data sent
10 between client 202 and server 204. Instead, password-based keys are used only
11 during login and other dynamically created keys are used to encrypt and decrypt
12 data sent across the network 210.

13 To be more precise, the key a principal uses to log in is really a hash of that
14 principal's password. A hash algorithm (sometimes called a message digest or a
15 checksum) produces a bit string that is a function of the information being hashed,
16 but that can't be used to recover the original input value. In other words, hashes
17 are one-way. Therefore, given a hashed password value, it is very nearly
18 impossible to recover the original password.

19 In certain authentication protocol supporting networks, users have
20 passwords, server applications that want to authenticate their users have
21 passwords, and all computers in a domain have passwords. Thus, any entity with a
22 password qualifies as a security principal.

23 One exemplary authentication protocol server itself, known as the Key
24 Distribution Center (KDC), runs on a domain controller where it has access to the
25 hashed password for every principal in its domain. This information is stored in a

1 directory associated with each principal, also kept on the domain controller.
2 Usually by default, the clear text password itself is not stored in the directory—
3 only a hashed version is kept there.

4 The authentication protocol allows negotiation of the encryption algorithm.
5 Most authentication protocol implementations default to a Data Encryption
6 Standard (DES) or the like, for example, wherein the keys have an effective length
7 of 56 bits. Some operating systems allow the authentication protocol to utilize a
8 stronger RC4 encryption algorithm.

9 That secret key encryption is used to send data privately is obvious, but it's
10 less than obvious how principles can use it for authentication, the authentication
11 protocol's most important function. To understand how this might be done,
12 imagine that two principles share a secret key with one another, and suppose the
13 first principle sends a message encrypted using that key. If the second principle
14 uses this key to decrypt the message, and that decryption works correctly, then this
15 message must have been encrypted using that same key. If only the two principles
16 know that key, then this message must have come from the first principle. Thus,
17 by proving knowledge of a key in this way, the first principle can be authenticated.

18 But this simple method is not very practical. Each set of principles (e.g.,
19 client/server pair) would have to share a secret key, which means that a separate
20 password for every service the client wanted to access securely. This is not a very
21 appealing prospect.

22 In accord with the exemplary authentication protocol, when a user wants to
23 prove their identity to a server application on some other system that user must
24 somehow provide the server with an appropriate ticket. Each ticket allows a
25 specific user to prove their identity to a specific server application, such as a

1 particular DCOM 206 application.

2 As graphically illustrated in Fig. 3, an authentication protocol ticket 240
3 contains both encrypted information 242 and unencrypted information 244. The
4 unencrypted part 244 of ticket 240, in this example, includes two primary pieces
5 of information: the name of the operating system "realm" or "domain", and the
6 name of the principal that the ticket identifies.

7 The encrypted part 242 of ticket 240 contains quite a bit more information.
8 In this example, the encrypted fields may include various flags, an encryption key
9 (commonly referred to as a session key) to be used later on, encrypted copies of
10 the user's principal name and domain, start and end times for this ticket's validity,
11 one or more IP addresses identifying the user's system, and the user's authorization
12 data, typically used by the server to determine what this user is allowed to access.

13 All of these fields are encrypted using the key of the server application this
14 ticket targets. Neither the user nor any attackers listening on the network can read
15 or modify the encrypted fields in a ticket, since they don't know the server
16 password used for encryption.

17 When a user wants to prove their identity to a server, they must acquire a
18 ticket to that server. In fact, virtually the entire exemplary authentication protocol
19 is devoted to acquiring and using tickets.

20 Before launching into a basic description of how the protocol works, it's
21 worth taking a moment to further explain the notation used below. Here, the
22 remaining text uses the following notation: K_X is the secret key (that is, the hashed
23 password) of X, K_C is the secret key of a client (C) user, K_S is the secret key of a
24 server (S) application, and K_K is the secret key of a KDC (K). Additionally, for
25

1 example, $\{\text{anything}\}K_X$ defines the “anything” as being encrypted with X's secret
2 key (i.e., K_X). Further let $\{T\}K_S$ be a ticket encrypted with server S's secret key
3 (i.e., K_S). In other words, this is a ticket for server S (the notation is a bit
4 imprecise, since the entire ticket isn't encrypted). Let, $K_{X,Y}$ be a session key used
5 between X and Y. Also, let $\{\text{anything}\}K_{X,Y}$ be “anything” encrypted with the
6 session key used between X and Y.

7 The first time a user requests a ticket is when that user logs in to some
8 account in an operating system domain, for example. From the point of view of
9 the user, the process is simple: type a login name, a domain name, and a password
10 into some client machine, then wait for the login to succeed or fail.

11 As shown in arrangement 300 in Fig. 4, the user's login request causes the
12 client system 302 to send a message 308 to a KDC 306 running on a domain
13 controller 304. The message 308 contains several things, including the user's
14 name; preauthentication data, which consists of a timestamp encrypted using K_C , a
15 hash of the user's password, as a key; and a request for a ticket-granting ticket
16 (TGT). The resulting TGT 310 is just an ordinary ticket, like the one shown in
17 Fig. 3, and as with all tickets, it is used to prove a user's identity. However, TGT
18 310 is used in a slightly special way in that the SSP on the client 302 presents it to
19 the KDC 306 when requesting tickets to specific server applications.

20 When request TGT message 308 arrives at domain controller 304, KDC
21 306 looks up the entry associated with the user's principal name in the specified
22 domain's directory database (not shown). From this entry, KDC 306 extracts a
23 hash of the user's password, and then uses this value to decrypt the preauthen-
24 tication data. If that decryption works and results in a very recent timestamp, then
25

1 KDC 306 can be certain that this user is who they claim to be, since the user has
2 demonstrated knowledge of the correct password. Note that this was done without
3 having that password sent over the network. To provide its services, the
4 exemplary authentication protocol never requires sending a user's password across
5 the network. If the decryption fails, the user must have entered the wrong
6 password, and the login will fail.

7 If the preauthentication data is correctly validated, KDC 306 next builds
8 and sends back to client machine 302 what it asked for, namely, a ticket granting
9 ticket (TGT) via message 310. Like all tickets, this one contains the user's name
10 and the name of the domain in which it was issued, along with a session key ($K_{C,K}$,
11 generated randomly by the KDC 306), the valid start and end times for this ticket,
12 and various flag values. Finally, in an Authorization Data field, the TGT contains
13 privileges and Security Identifiers (SIDs), or the like; essentially identifying this
14 user and the groups the user belongs to (e.g., these may be extracted from the
15 user's entry in a directory). As before, part of the ticket is encrypted using the key
16 of the server to which this ticket will be sent. Since the TGT is used only to
17 request other tickets, and since only the KDC can give out tickets, the encrypted
18 part of the TGT is encrypted using K_K , the key of the KDC itself.

19 Along with the TGT, the KDC also sends back to the client machine the
20 session key $K_{C,K}$, the same value the server placed in the TGT. This session key is
21 sent encrypted using the user's hashed password as a key. When the client system
22 gets message 310, it uses the hash of whatever password the user has entered to
23 decrypt the received session key. In certain implementations of the exemplary
24 authentication protocol, this decryption will always work, since only users who
25 demonstrate knowledge of the correct password via the preauthentication data will

1 get message 310 sent to them at all. Sending preauthentication data is optional in
2 the authentication protocol standard.

3 Once a user/client has successfully logged in, they will likely begin
4 accessing services running on other computers in the network. To do this securely,
5 the user must at a minimum have some way of proving their identity to those
6 services. As shown in Fig. 5, the SSP can be used to present a TGT to the KDC,
7 requesting a ticket to a specific service. Here, via message 314, client machine
8 302 is requesting a ticket for server machine S (312). To distinguish them from
9 TGTs, these tickets are sometimes called service tickets, but the format is identical
10 for both types. That ticket is then sent to the target service (via message 306),
11 which can use it to determine exactly who this user is. In fact, immediately after
12 acquiring a TGT, most clients typically complete the login process by requesting a
13 service ticket for its own computer, allowing it to prove its identity to local
14 services.

15 When a user wants to access a DCOM server application (called, for
16 example, Server S) running on some remote system 312, the user will load the
17 client part of the application, and this client will attempt to create a remote DCOM
18 object (not shown). If the application uses the authentication protocol for
19 authentication, that client application will need to acquire a ticket on behalf of its
20 user before it can access the server. Recall that each ticket authenticates a
21 particular user to a particular service, and since the client part of a distributed
22 application runs on behalf of the user, that client acquires tickets that prove the
23 user's identity to the server.

24 When the client application makes its first remote request to the server, a
25 ticket request message 308 is automatically made to KDC 306, as shown in Fig. 5.

1 Although not shown, ticket request message 308 contains several things, including
2 the user's TGT, the name of the server application for which a service ticket is
3 requested (which in this case is Server S), and an authenticator proving that this
4 TGT belongs to this user. The authenticator may include, for example, the current
5 time and the user's name, and is encrypted using the session key $K_{C,K}$ that was
6 received at login.

7 When KDC 306 receives ticket request message 308, it decrypts the TGT
8 (recall that only the KDC knows K_K , the key used to encrypt this ticket), then
9 extracts the session key $K_{C,K}$ from the ticket. KDC 306 then uses this session key
10 to decrypt the authenticator. The authenticator serves two purposes. First, because
11 it is encrypted using the client/authentication protocol session key, it proves that
12 the user is who they claim to be, since as described earlier, the only way to get this
13 session key is to type the correct password at login. If the KDC's attempted
14 decryption of the authenticator is successful, client 302 must be in possession of
15 the session key.

16 Secondly, because the authenticator contains a timestamp, it significantly
17 prevents an attacker from grabbing a user's TGT off the network, then presenting it
18 as its own. A new authenticator is created each time a ticket is used, and because
19 the timestamp is encrypted using the session key, known only to client 302 and
20 KDC 306, a valid authenticator cannot be created by anyone else.

21 To prevent resending authenticators, KDC 306 will reject any authenticator
22 whose timestamp is too old. By default, for example, in certain systems an
23 authenticator's timestamp must differ by no more than 5 minutes from that of the
24 server that receives it. This implies that the clocks on machines using the
25

1 authentication protocol must be at least loosely synchronized. Thus, for example,
2 an IETF-defined Simple Network Time Protocol (SNTP) or the like may be used
3 for clock synchronization. To further ensure that the user isn't presenting a stolen
4 ticket, KDC 306 may also verify that the IP address in the TGT matches the IP
5 address of the system that sent this ticket request message 308.

6 If everything checks out, KDC 306 will believe that this user is who they
7 claim to be, and will send back the requested service ticket through message 310.
8 KDC 306 copies some fields from the TGT into the new service ticket, including,
9 for example, the user's name, domain, and authorization data. It sets the service
10 ticket's flags and start/end times appropriately, and generates a new random
11 session key, $K_{C,S}$, which it places in the ticket. As described later, this key can be
12 used to encrypt information sent between client machine 302 and Server S 312.
13 KDC 306 then encrypts this new ticket using K_S , Server S's secret key, and sends it
14 back to client machine 302, along with the new session key $K_{C,S}$. To prevent
15 attackers from learning this new key, it can, for example, be sent encrypted using
16 the session key shared by the authentication protocol and the client.

17 Finally, the goal of this entire exercise can be achieved as the user proves
18 their identity to the server application. On its first request to Server S 312, client
19 302 presents the service ticket it just received along with an authenticator
20 encrypted using $K_{C,S}$. This is shown in message 318. This information is sent as
21 part of whatever protocol is being used between client and server. With DCOM,
22 for example, the ticket and authenticator may be carried in a particular field in the
23 appropriate DCOM packet. The receiving system decrypts the ticket with its
24 secret key, extracts the session key $K_{C,S}$ from the ticket, then decrypts and
25

1 validates the authenticator. If everything checks out, the user's identity
2 information is extracted from the ticket—e.g., principal name, domain name, and
3 authorization data—and made accessible to the server application. The SSP
4 usually does most if not all of this work.

5 Although the authentication protocol itself does not directly address the
6 problem, the information about the user that is extracted from the received ticket
7 can eventually be used to make an authorization decision. Exactly how this is
8 done is up to the creator of the server application. It might look up the user's name
9 in a list of users authorized to perform some function (e.g., read or write), or it
10 might use the authorization data to impersonate the user (e.g. proxy), for example.
11 In this second case, the Local Security Authority (LSA) on the server's machine
12 can, for example, construct a security access token using the user's authorization
13 data. Once this is completed, the server process may use this token to impersonate
14 the user and try to access whatever resource the user is interested in.

15 Thus, how an authorization decision is made is usually not within the
16 authentication protocol's purview, but the exemplary authentication protocol does
17 guarantee that the identity the user is claiming in this service ticket truly identifies
18 that user.

19 To summarize, since the service ticket the user presented was encrypted
20 using Server S's secret key, and since only KDC 306 (along with Server S, of
21 course) knows that key, this ticket must have been created by the KDC. Since
22 KDC 306 will only give out service tickets to users who can prove they know the
23 right password by correctly encrypting the preauthentication data, this user must
24 be who they claim to be. When presented with valid authenticators, the tickets
25

1 that are used by the authentication protocol to provide reliable authentication of
2 clients.

3 It might be possible for an attacker to install a spurious version of Server S,
4 then acquire sensitive information by fooling client 302 into thinking it was the
5 real Server S 312. To prevent this, the exemplary authentication protocol standard
6 defines an option for mutual authentication, an option that should be requested by
7 most applications. Not only does the client user prove its identity to the server, but
8 the server must also prove its identity to the client.

9 To do this, the SSP on the server creates a message containing the
10 timestamp from the client's authenticator encrypted using the client/Server S
11 session key, $K_{C,S}$. When the SSP on the client receives this message, it can then
12 use its copy of the session key to decrypt it. If the client's SSP finds the timestamp
13 it just sent, it can be further certain that the server knows the session key, too.
14 Since learning the session key required decrypting the server's ticket, which
15 required knowing the server's password, then this server must be who it claims to
16 be.

17 All of the complexity described so far has focused on how the
18 authentication protocol provides just one security service, namely authentication.
19 But the exemplary authentication protocol can also provide data integrity and data
20 privacy, two other useful services. Because the exchanges just described have left
21 the client and server in possession of a shared session key, providing these
22 additional services is straightforward.

23 For example, to prevent an attacker from modifying transmitted data
24 without being detected, the SSP on any system that's sending data can compute a
25

1 checksum on each packet it sends and transmit that checksum with the packet.
2 The checksum value is a function of the data it's based on, so if the data is
3 changed, the checksum must also change. But sending just a packet and a
4 checksum isn't always sufficient since an attacker could grab the packet, modify
5 the data, recompute a new checksum on the new data, and send it on its way.

6 To prevent this from happening, the data's sender may compute the
7 checksum on not just the message itself, but on the message and other information,
8 then encrypts the result using the session key. By default, the checksum algorithm
9 used in certain exemplary authentication protocol arrangements is termed HMAC
10 (Hash-based Message Authentication Code), and the checksum is encrypted using
11 RC4 or the like. An attacker will be unable to create a valid checksum for
12 modified data, since they do not know the key. The result is that the receiver of a
13 packet can always detect any attempt to modify the contents of that packet.

14 Providing the last service, data privacy, is simple. Since the client and
15 server share a session key, the SSP on each one just uses this key to encrypt data it
16 sends to the other. Note that data privacy implies data integrity, since no attacker
17 can modify encrypted data in transit on the network without those changes being
18 detected.

19 In this manner, the exemplary authentication protocol provides the
20 fundamental security services required for a distributed environment:
21 authentication, data integrity, and data privacy. The authentication protocol may
22 also be configured to support delegation.

23 In the example described earlier, suppose the user has already been
24 authenticated to Server S 312. Server S 312 can now impersonate the user and
25

1 attempt to access something on its local system, such as a file. In this case, the
2 access checking built into the operating system (or like program) will grant or
3 deny the access based on the file's access control list (ACL) or like mechanism.
4 All of this works naturally if the resource being accessed is on the same machine
5 as the server.

6 Suppose, however, that to carry out whatever task the user is requesting,
7 Server S must make a request to another server, e.g., Server T 328, running on
8 another machine (see Fig. 6). Even though Server T's direct user will be Server S
9 312, access is being requested on behalf of the original user (i.e., client 302), not
10 Server S 312. For this to work correctly, the user needs some way to pass its
11 identity to Server S, allowing Server S to make further remote requests on its
12 behalf.

13 The exemplary authentication protocol supports this concept through
14 delegation, as shown in Fig. 6. If a client application requests it, a user's TGT and
15 its associated session key can be passed to another server, such as Server S.
16 Sending just the TGT wouldn't be enough, since the associated session key is
17 required to construct the authenticators sent along with the TGT when new tickets
18 are requested. Like all tickets, the TGT is encrypted, but to ensure that attackers
19 can't steal the TGT's associated session key off the wire when it's passed from
20 client to server, that key is sent encrypted using the session key the client shares
21 with Server S.

22 The TGT passed by client 302 to Server S 312 must have the
23 FORWARDABLE flag set in its Flags field. If it does, Server S can present this
24 TGT to a KDC and request tickets to other services even though the IP address in
25

1 this TGT won't match Server S's IP address—the FORWARDABLE flag tells the
2 KDC that it's okay to ignore this discrepancy. Also, in certain implementations
3 client 302 can only send its TGT and associated session key to a server if that
4 server's account is marked as trusted for delegation in this domain.

5 To see how this all works, suppose client 302 passes its TGT and associated
6 session key to Server S 312 via message 320 with the FORWARDABLE flag set.
7 If Server S 312 needs to access Server T 328 on the client user's behalf, it can
8 present this TGT along with a valid authenticator to KDC 306 via message 322,
9 requesting a new ticket for Server T 328. This new ticket will contain the user's
10 identity, just like the original ticket, but will be encrypted using Server T's
11 password (see message 324). When Server S 312 presents this new ticket to
12 Server T 328 via message 326, again with a valid authenticator, Server T 328 will
13 behave as though it is receiving a request directly from client 302.

14 Kerberos also allows authentication between clients and servers in different
15 domains, although the process is a bit more complex than that described so far. To
16 authenticate itself to any server application, no matter what domain that server is
17 in, a user must acquire a ticket to that server. But only a KDC 306 in the same
18 domain as the target server can issue that ticket, since only it knows that server's
19 password. If a user wants to be authenticated to a server in a different domain,
20 then they must request a ticket to that server from a KDC in the foreign domain.
21 As is always the case, requesting a ticket from a KDC requires presenting a TGT
22 to that server. Thus, the fundamental problem is for the user to acquire a TGT to
23 the KDC in the foreign domain. Once the user has this, they can request and use a
24 ticket to the target server in the normal way.

1 For this to be possible, the two domains must have a trust relationship
2 between them. When a trust relationship is created between two domains, a
3 password is also created that is known only to those two domains. This shared
4 password can be used to encrypt a ticket that's passed between the two domains.

5 Fig. 7 shows how the process works (and although they have been omitted
6 from the diagram for simplicity, authenticators and session keys are still used as
7 described earlier). Suppose a user in the acct.acme.com domain 400 wants to ac-
8 cess Server Q 403 in the sales.acme.com domain 402. The Kerberos SSP on the
9 client system 302 begins by presenting, via message 404, the user's TGT to the
10 domain controller 304(a) (and KDC 306(a) therein) within the user's own
11 acct.acme.com domain 400, requesting a TGT to the KDC 306(b) in domain
12 controller 304(b) of sales.acme.com domain 402. KDC 306(a) responds in
13 message 406 by sending back a TGT that client 302 can use to request tickets from
14 the KDC 306(b) in the sales.acme.com domain 402. This ticket is not encrypted
15 using the password of this domain's KDC 306(b), as it normally would be. Instead,
16 it's encrypted using the password shared between acct.acme.com domain 400 and
17 sales.acme.com domain 402, designated K_X in Fig. 7.

18 Once it has this TGT, the client's SSP then presents it to KDC 306(b) in
19 sales.acme.com domain 402, requesting a ticket for Server Q 403 (as represented
20 by message 408). KDC 306(b) then decrypts the TGT presented by client 302
21 using K_X , the password it shares with acct.acme.com's KDC 306(a). If the TGT is
22 valid, then KDC 306(b) looks up Server Q's password in its directory database and
23 uses it to build a ticket for Server Q. It then sends this ticket back to the client 302
24 via message 410, which can subsequently present it to Server Q 403 as shown by
25

1 message 412.

2 Despite the apparent complexity of Fig. 7, the mechanics of using the
3 authentication protocol between domains are simple. All that's required is adding
4 the single extra step of getting a TGT for a foreign domain's KDC. But think
5 about what the KDC in sales.acme.com is implicitly assuming in this example: it's
6 trusting the KDC in acct.acme.com not to give out TGTs to it without first
7 validating the user's identity. In other words, by accepting a TGT encrypted with
8 the key it shares with acct.acme.com, sales.acme.com's KDC is trusting the KDC
9 in acct.acme.com to behave correctly. For cross-domain authentication to work,
10 the two domains involved must trust each other.

11 The exemplary authentication protocol also supports transitive trust, in
12 which if one domain trusts a second domain, and if that second domain trusts a
13 third domain, then there is automatically a trust relationship between the first
14 domain and the third domain. Transitive trust ensures that domains only need to
15 share a password with the domains immediately above and below them in the
16 domain hierarchy—the authentication protocol takes care of the rest.

17 It is further relevant to this description to note that the authorization portion
18 of a ticket may optionally include an optimization data field suitable for passing
19 application specific data. Here, for example, Windows® client-server software
20 available from Microsoft ® Corporation of Redmond, Washington, takes
21 advantage of this optional field by providing additional data about the client/user.

22 As networks grow in size and complexity, users will be provided with ever-
23 greater services and tools. For example, e-commerce on the World Wide Web
24 continues to grow and expand. Consumers are able to shop for and purchase
25

1 goods and services directly from their providers, or from third parties. One recent
2 move has been to provide one-stop shopping websites or portals through which
3 consumers conduct most if not all of their on-line business.

4 These websites and others tend to use a hierarchical server structure that
5 includes at least one front-end service and at least one back-end service, for
6 example. Moreover, since many of these services/servers are interested in the user
7 (e.g., a potential or actual customer) it is not uncommon for the authorization data
8 portion of a ticket (see, e.g., Fig. 3) to include privileged client information of
9 interest to different parties. For example, a Privilege Attribute Certificate (PAC)
10 or the like may be included within the authorization data portion of a ticket to
11 further streamline the client-server environment. In a multiple server arrangement
12 a front-end server may forward the ticket on through to a subsequent
13 service/server, as shown in Fig 8.

14 In this example, the first server may be directed to forward a ticket to one
15 of the other servers. In certain systems, to access more than one server, the client
16 will need additional tickets. In other systems, "server 1" would decrypt the ticket
17 and then duplicate and pass on the user information to one or more of the other
18 servers. These techniques and others like them tend to consume significant
19 amounts bandwidth, memory and processing resources. This is especially true for
20 aggregated service websites on the Internet. Moreover, the user may not have
21 meant to authorize the duplication and/or release of their privileged information to
22 so many entities.

23 One example of an aggregated service website is an Acme Travel Company
24 website, which accepts user inputs about travel plans and then attempts to match
25

1 the user to prospective goods/service providers, such as, airlines, railways, hotels,
2 rental car agencies, etc. Here, in the past, let us assume that the Acme server
3 would decrypt the ticket and then possibly provide duplicates of the PAC or other
4 portion of the ticket to a United Airlines server, an American Airlines server, and a
5 Not-So-Good Airline server. While the user may be willing to fly on United
6 Airlines or on American Airlines, they may not want to fly on Not-So-Good
7 Airline, nor may they want to have their privileged information provided to Not-
8 So-Good Airlines.

9 Unfortunately, the client is at the mercy of the decrypting server to enforce
10 certain policies since the existing authentication schemes do not allow for
11 selective forwarding on the part of the client.

12 Having recognized this problem and anticipating the future trends in
13 website businesses and arrangements, in accordance with certain aspects of the
14 present invention, the client is provided with the means to selectively control who
15 is allowed to access privileged and other profile information within a ticket or like
16 security token. As described below, it should be recognized that authorization data
17 may be provided and later used via the ticket and/or an authenticator (e.g., through
18 an optional field). The various methods and arrangements allow for information,
19 such as the profile, to be encoded in such a way that only a subset of parties (e.g.,
20 servers/services) can access the data, regardless as to where the information may
21 be forwarded.

22 More specifically, for example, let P be the profile or other information
23 supplied by the client within the ticket. The client constructs a ticket with P being
24 encrypted with a randomly generated key, K_p . This is noted by $\{P\}_{K_p}$, which
25

1 includes a single encryption of the P information. The client then, either
2 independently or with the assistance of a public, or other trusted third party
3 authority, appends $\{K_p, \text{nonce}\}K_{id1}$, $\{K_p, \text{nonce}\}K_{id2}$, ..., $\{K_p, \text{nonce}\}K_{idn^{th}}$ to
4 $\{P\}K_p$, and supplies it to the intermediary servers/services. Where now, no matter
5 who obtains the forwarded ticket, they cannot decrypt P without having an
6 appropriate key K_{idx} (wherein x is 1, ..., n) required to obtain key K_p . Note that
7 the nonce data may be optional, depending upon the implementation.

8 Note that key K_{idx} can be assigned to a specific group(s) or an individual,
9 for example. In certain cases, such an assignment may minimize the amount of
10 data that is to be stored, processed or transmitted. A third party authority can
11 supply value by providing interesting grouping of these end services, and handle
12 the registration and updates of any keys.

13 Fig. 9 illustrates a modified portion of an exemplary ticket 500 further
14 having at least one encrypted client information portion 502 within the
15 authorization data field. Here, for example, the encrypted client information
16 portion 502 includes privileged user information P that is encrypted with key K_p .
17 As can be seen, a server/service with access to K_{id1} may decrypt $\{P\}K_p$ by first
18 decrypting K_p using K_{id1} . Likewise, a server/service with access to K_{id2} may
19 decrypt $\{P\}K_p$ by first decrypting K_p using K_{id2} . Any number of such embedded
20 key arrangements may be realized.

21 Thus, in the example above, the user may specifically/intentionally leave
22 out an encrypted key for Not-So-Good Airlines while including one or more
23 associated with United Airlines and American Airlines. Any arbitrary grouping
24 may also be applied by selectively controlling the keys and the addition of the
25

1 encrypted keys to the modified ticket 500.

2 Those skilled in the art will recognize that these improved methods and
3 arrangements may, for example, be implemented using the above-described
4 exemplary authentication protocol (e.g., Kerberos), or other authentication
5 protocol(s). Thus, although some preferred embodiments of the various methods
6 and arrangements of the present invention have been illustrated in the
7 accompanying Drawings and described in the foregoing Detailed Description, it
8 will be understood that the invention is not limited to the exemplary embodiments
9 disclosed, but is capable of numerous rearrangements, modifications and
10 substitutions without departing from the spirit of the invention as set forth and
11 defined by the following claims.